The BeamerSubFrame Package

Reordering frames in the PDF file without reordering the source

Mike Kaufmann
m.km@gmx.de

2011/08/07 (v0.2)

Abstract

The BeamerSubFrame package provides a method to reorder frames in the PDF file without reordering the source. Mainly, it is meant to embed or append frames with details on some subject.

Contents

1 Introduction

1.1 Why this Package

Of course, usually a presentation is prepared for one occasion. But sometimes a presentation is used more then once, and the audience and/or the available time differs.

For me, there are two kinds of audience. The first group are the sales people, who for the most part only want to know the features of a new product I developed. And the second group are the technicians, who want to know everything.

So basically, on one time I have to give a presentation in half an hour, covering only the basics. But there might be questions, where I need to show some details. In this case it's nice to have the necessary frames. And an another time, I have to give a presentation in one hour, covering also the details.

In both cases, it would be nice to have a presentation, where normally only the cursor keys are needed for navigation, and without the necessity of clicking on links.

With Beamer, one would have to create two presentation. One with the frames containing details at the end, and another one with the details between the other frames.

Now here the BeamerSubFrame package comes in handy. After some changes to the source of the presentation with the details embedded, both version can be compiled out of the same source. To select the version, only one option must be changed.

1.2 A Warning

This package is in an early state of development (it's version 0.2). It is already useful, but there are some restrictions (see section 5). And although it's not planned, the user interface may change in the future.

1.3 Feeback and Testing

Because this package is very new and still under development, any feedback is appreciated. A good location for a discussion would be the newsgroup `comp.text.tex` (or `de.comp.text.tex`).

And since Beamer offers a lot of features and it is hard to test the BeamerSubFrame package with all of them, it would be nice, if some people would do some additional testing and/or provide some presentation for testing.

1.4 Dependencies

The BeamerSubFrame package can only be used with the Beamer class (article mode is not supported yet).

Additionally, it needs the `verbatim` package.

1.5 Legal Stuff

2 Using the Packages

2.1 Package Options

The BeamerSubFrame package has three options.

`embed` The option `embed` is used to put frames with details between the other frames.

`append` The option `append` is used to remove the frames with details from the other frames and append them at the end of the PDF file.

If none of these two options is given, `embed` is used as default. Both options are mutually exclusive. If both are given, the last one wins.

`nominiframes` In themes with miniframes, there are also frame symbols for appended frames with details by default. If the option `nominiframes` is given, these frame symbols will disappear. This option has only an effect, if the `append` option is given too.

2.2 The `subframe` Environment

`subframe` For frames which should be embedded or appended contitionally (i.e. frames with details), the `subframe` environment must be used instead of the `frame` environment. Basically, it works the same way as the `frame` environment, i.e. it has the same options and arguments.

For typesetting, the `frame` environment is used internally. In embed mode (i.e. the `embed` option is given) it inserts the option `environment=subframe` to enable the use of verbatim material in the frames. In append mode the contents of the frame is written to a file. Before the contents, `\begin{frame}`, with all arguments of the `subframe` environment appended, is inserted. And after the contents `\end{frame}` is inserted.

For copying the contents, the `verbatim` package is used. So in append mode the `subframe` environment is treated as a `verbatim` environment,[1] but without typesetting anything.

There should be no parts, sections, or subsections only containing subframes, because in append mode, they would have no contents. Since it is not the intention of the BeamerSubFrame package, to move parts, sections, or subsections, this was not even tested.

2.3 Appending Frames

`\appendsubframes` To append the subframes, the macro `\appendsubframes` must be called before `\end{document}`. In embed mode, the command simply does nothing. In append mode it inputs the file written by the `subframe` environments.

The command is only allowed once and it can only appear right before `\end{document}`.[2] Disregarding this may lead to some strange errors.

The command can be omitted to remove the subframes from the PDF file in append mode. But links to them will then produce warnigs and will lead somewhere else. In themes with miniframes, the links of the miniframe symbols for subframes will lead to page 1.

2.4 Conditional Execution

`\ifappend` To execute some macros or insert some text according to the mode of the BeamerSubFrame package, the marco `\ifappend` can be used. It has two arguments.

`\ifappend{`⟨*material for append mode*⟩`}{`⟨*material for embed mode*⟩`}`

[1] Because of this, the phrase `\end{subframe}` cannot be used in a `subframe` environment.
[2] Both restrictions are not tested by the package yet.

The first argument is inserted only in append mode and the second one only in embed mode.

The macro is intended for conditional links in frames and subframes. It can be used in own macros. To do something only in one mode, the argument for the other mode can be left empty.

2.5 The `lastframe` Environment

`lastframe` Normally, a presentation has a last frame, containing something like "Thanks for listening" (at least I learned it this way). But using a `frame` environmet, the navigation (sidebar, miniframes, and so on) appears as if the frame belongs to the last (sub)section.

Using the `lastframe` environment, the navigation has the same appearance as on the title frame.

It has the same options and arguments as the `frame` environment, which is used internally with all given arguments.. To enable the use of verbatim contents, it additionally inserts the option `environment=lastframe`.

The `lastframe` environment must be used only for the last frame (before the `\appendsubframes` command), because it sets the counters for parts, sections, and subsections to 0. So after a "lastframe", sectioning will not work properly anymore.

2.6 Inserts

`\inserttotalframenumber` Beamers `\inserttotalframenumber` was changed. In append mode it now contains the number of frames without the subframes. In embed mode it contains the number of frames as usual.

`\inserttotalframenumberwithsub` Additionally, the insert `\inserttotalframenumberwithsub` is available. It inserts the total number of frames including appended subframes in append mode. It is also defined in embed mode. Then it just contains the total number of frames (so the same as `\inserttotalframenumber`).

2.7 Files written

The BeamerSubFrame package writes two files.

`\jobname.sfr` contains the contents of all subframes. The package inputs this file before the end of the document with the `\appendsubframes` command. The file is written regardless of `\nofiles`.

`\jobname.sfp` contains the information necessary for miniframes. It is written indirectly, using the `.aux` file. So if `\nofiles` is used, the `.sfp` file is not written too.

Both files are only written and used in append mode.

2.8 Warnings and Errors

Currently, there is one warning and one error in the BeamerSubFrame package.

When loading the package, it checks, if the Beamer class is loaded. If not, an error message will be displayed and the package is not loaded. Disregarding this error will therefore lead to additional errors.

The `.sfp` file is checked at the end of the document. If it is missing, corrupted or incomplete, a warning is given out. In this case some links of miniframe symbols for subframes will lead to page 1. Of course this applies only to themes with miniframes.

2.9 Conditional Execution (2)

`\ifsubframe` This macro is intended to be used in themes, to make things appear differently on normal frames and subframes. This could be used to set some marker on subframes.[3] The command works regardless of the mode.

`\ifsubframe{`⟨*material for subframes*⟩`}{`⟨*material for normal frames*⟩`}`

The first argument is inserted only in subframes and the second one only in normal frames. An unused argument may be left empty.

2.10 The `subslideentry`

`\subslideentry` This is not a macro for normal users. It may be useful for people who want to write their own theme, willing to do more, then defining and using templates. If you are not one of these people, just skip this section.

In append mode, the BeamerSubFrame package writes a `\subslideentry` instead of a `\slideentry` for subframes to the `.nav` file. The `\subslideentry` has the same arguments as the `\slideentry`. By default, it just passes its arguments to `\slideentry`. This way it does the same as `\slideentry`, independent of the used theme. By redefining `\subslideentry`, it is possible to, for example, make miniframes for subframes appear differently form miniframes for normal frames.

3 The Look and Feel

3.1 In Embed Mode

In embed mode, the frames containing details are between the other frames. And navigation (navigation bar, sidebar and so on) behaves the same way, as it would without the BeamerSubFrame package. Of course the highlighting of section and subsection names is the same too.

[3] There is an example in the example file, which can be generated from `beamersubframe.dtx`.

3.2 In Append Mode

In apend mode, the frames containing details are at the end of the PDF file.

The navigation in the main part (the part of the PDF without the details) behaves as if the frames with details are not part of the PDF. And for the total number of frames, the frames with details are not counted (leading to something like "41 / 29" for the subframes in themes with a footline like AnnArbor).

In themes with miniframes there are also frame symbols for the frames with details. Clicking on the symbols leads to these frames. But if the option `nominiframes` is given, the frame symbols will disappear and the symbol for the normal frame before the frame with details is highlighted on the frame with details.

The navigation bar in the appended part (the part of the PDF with frames containing details) behaves as follows

slide and frame symbol: behave as if the appended frames where at the end of the source, so leading to the next or previous slide or frame in the PDF (arrows), or leading to the last or first slide of a frame.

section and subsection symbol: are always leading to the appropriate slides in the main part of the PDF.

presentation symbol: leads to the first frame of the presentation or the last frame of the main part (before a possible appendix).

appendix symbol: leads to the first or last frame of the appendix.

The links for sections and subsections in the sidebar and others lead to the first slide of the section or subsetion.

In the navigation part of appended frames, the section and subsection names are the same and the highlighting is done as if the frames where embedded.

4 A practical Approach

4.1 The Order of Frames

If a presentation contains details on some items, there are two ways to insert them.

1. The details can be inserted right after the according item. This is useful, if it is planned to present the details.

2. The details can be inserted at the end. This is useful, if it is not planned to present the details, but they might be needed to answer question.

In both cases, links can be useful. In the first case, these are links to skip the details. And in the second case, these are links to jump to and return from the details.

The first case is shown in Figure 1 and the second in Figure 2. In both pictures the black lines denote the transitions using the cursor keys, and the blue lines the

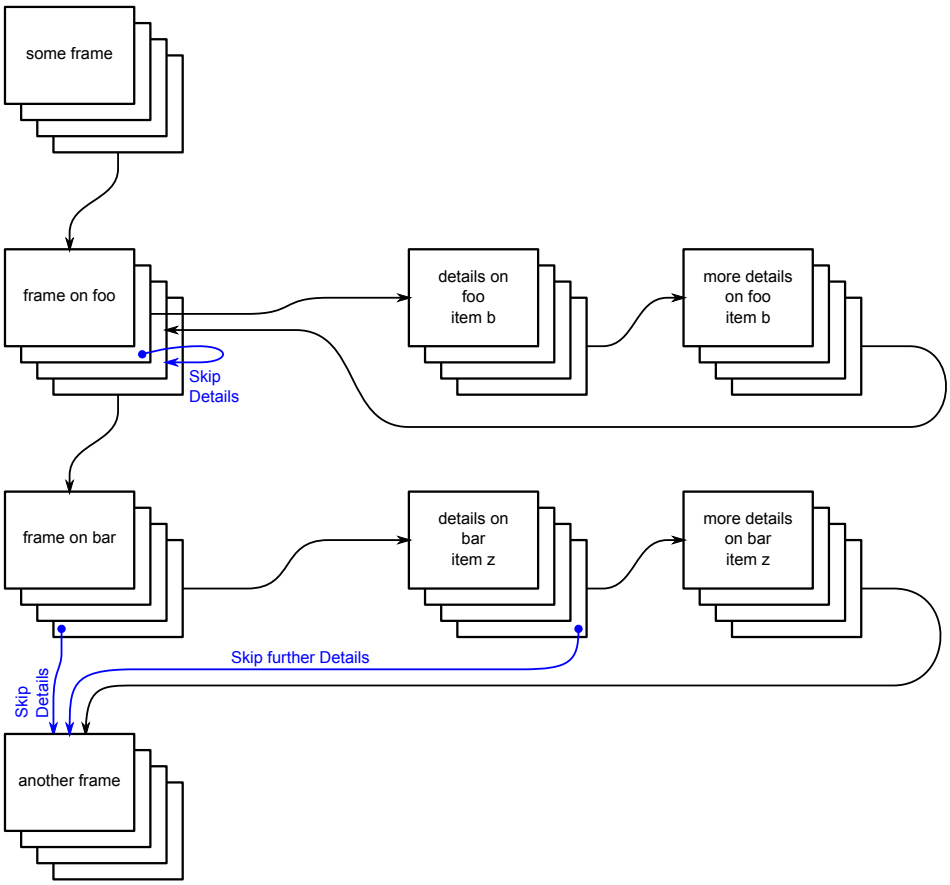

Figure 1: Frame Order with Details embedded

transitions using links. The dashed black lines are transitions by cursor keys, which are normally not needed.

Of course, the way the links are arranged in the picuters is only an example. There are hundreds of possibilities. And how the links are arranged the best way, depends on the presentation *and* on the person giving the presentation. For this section, let's consider the links in the pictures the ideal way.

To use the BeamerSubFrame package, in the source the frames must be arranged as shown in Figure 1. By default, the frames in the PDF file are arranged the same way. By just using the option `append`, the frames in the PDF file will be arragned as shown in Figure 2.

4.2 Basic Changes to the Source

To use the BeamerSubFrame package, a few changes to the source of a presentaion must be applied. In the example below, the left side shows the source before the

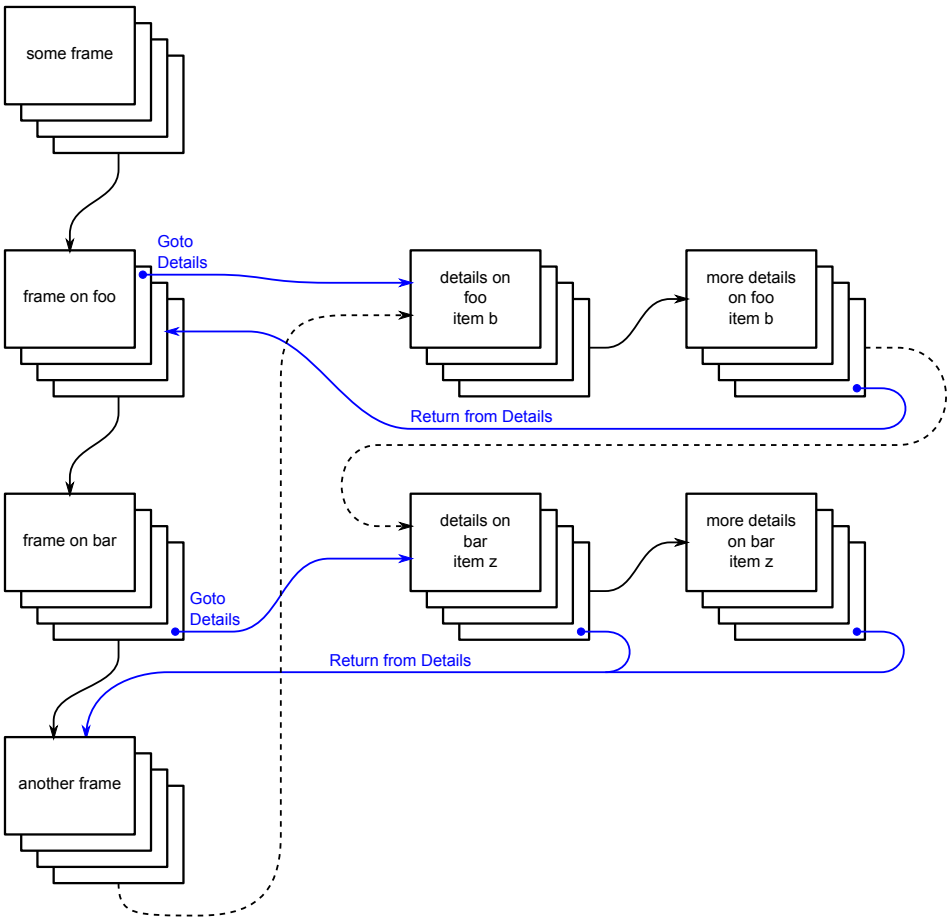

Figure 2: Frame Order with Details appended

changes and the right side the source after them.

```
\begin{frame}{Frame on Foo}
material on foo
\end{frame}

\begin{frame}
    {Details on Foo Item b}
details for item b
\end{frame}

\begin{frame}{Another Frame}
other material
\end{frame}

\end{document}
```

```
\begin{frame}{Frame on Foo}
material on foo
\end{frame}

\begin{subframe}
    {Details on Foo Item b}
details for item b
\end{subframe}

\begin{frame}{Another Frame}
other material
\end{frame}
\appendsubframes
\end{document}
```

Here, the `frame` environment for the frame with details was changed to the `subframe` environment, and the command `\appendsubframes` was inserted before `\end{document}`. This is the minimum of changes necessary.

It is not necessary, to change the arguments of subframes. Also, there is no need to change the contents of subframes. Even the `fragile` option (for verbatim material) will work, because the `subframe` environment automatically adds the option `environment=subframe`.

4.3 Conditional Links

How links are used and arranged, it totally up to the user, because this is highly individual. But nevertheless, links in embed mode should be different from links in append mode. For example, a link on a frame before a subframe (or a group of subframes) with details should skip them in embed mode, while in append mode the same link should lead to the subframe (or the first subframe of the group).

Let's stay with the examples shown in Figure 1 and Figure 2.

The link "Skip Details" on slide 2 of "frame on foo" in embed mode is replaced by a link "Goto Details" in append mode. In the first case the destination is slide 3 of "frame on foo" and in the second case it is slide 1 of "details on foo item b".

Let's say, "frame on foo" has a label "lfoo", given as option `label=lfoo` to the `frame` environment (this is also necessary, if the BeamerSubFrame package is not used). For append mode a label must be added to the subframe "details on foo item b". Let's call it "ldfoob".

For embed mode the link can be realised with

```
\uncover<2>{\hyperlink{lfoo<3>}{\beamerskipbutton{Skip Details}}}
```

and for append mode with

```
\uncover<2>{\hyperlink{ldfoob}{\beamergotobutton{Goto Details}}}
```

In order to make the link work as expected in both modes, without the need to change the source, `\ifappend` can be used like this:

```
\ifappend{\uncover<2>{\hyperlink{ldfoob}%
                        {\beamergotobutton{Goto Details}}}}%
         {\uncover<2>{\hyperlink{lfoo<3>}%
                        {\beamerskipbutton{Skip Details}}}}
```

The link to return from the frames with details can then be done this way:

```
\ifappend{\uncover<4>{\hyperlink{lfoo<3>}%
                        {\beamerreturnbutton{Return from Details}}}}{}
```

This has the disatvantage, that in embed mode there is no link, and therefore the layout will differ between embed mode and append mode. But this can be fixed by adding an invisible button like this:

```
\ifappend{\uncover<4>{\hyperlink{lfoo<3>}
                        {\beamerreturnbutton{Return from Details}}}}%
         {\visible<0>{\beamerreturnbutton{Return from Details}}}
```

Typing all this every time can be a bit exhausting. So let's define some macros.

For skipping or going to the details, a macro called `\linkdetail` can be defined as

```
\newcommand<>{\linkdetail}[4]{%
    \ifappend{\uncover#5{\hyperlink{#3}{\beamergotobutton{#4}}}}%
             {\uncover#5{\hyperlink{#1}{\beamerskipbutton{#2}}}}
}
```

The command is overlay-aware and it has four arguments:

`#1` is the label for the destination to go to in order to skip the details.

`#2` is the text for the link to skip the details.

`#3` is the label for the destination to go to in order to jump to the details.

`#4` is the text for the link to go to the details.

Now the link on the frame before the subframe can be written as

```
\linkdetails<2>{lfoo<3>}{Skip Details}{ldfoob}{Goto Details}
```

For returning from the details, a macro called `\returndetail` can be defined as

```
\newcommand<>{\returndetail}[2]{%
\ifappend{\uncover#3{\hyperlink{#1}{\beamerreturnbutton{#2}}}}%
         {\visible<0>{\beamerreturnbutton{#2}}}
}
```

The command is also overlay-aware and it has two arguments:

`#1` is the label for the destination to go to in order to return from details.

`#2` is the text for the link to return from details.

And the link on the subframe can be written as

```
\returndetails<4>{lfoo<3>}{Return from Details}
```

Of course, these are only examples. How links are done, is up to the user. And therefore, it is also up to the user, how commands like `\linkdetail` and `\returndetail` are defined.

Another example: I like my links to be active only on the slide, they are needed. On all other slides, they should be covered and inactive. So I definded the macros a bit different:

```
\newcommand<>{\linkdetail}[4]{%
    \ifappend{\uncover#5{\alt#5{\hyperlink{#3}{\beamergotobutton{#4}}}%
                                {\beamergotobutton{#4}}}%
             {\uncover#5{\alt#5{\hyperlink{#1}{\beamerskipbutton{#2}}}%
                                {\beamerskipbutton{#2}}}}
}
\newcommand<>{\returndetail}[2]{%
    \ifappend{\uncover#3{\alt#3{\hyperlink{#1}{\beamerreturnbutton{#2}}}%
                                {\beamerreturnbutton{#2}}}}{}
}
```

4.4 An Example

4.4.1 Without BeamerSubFrame

As an example, let's realise frames in the order shown in Figure 1 and Figure 2. To show the difference of the source with and without using the BeamerSubFrame package, first the order shown in Figure 1 is realised without BeamerSubFrame.

```
\documentclass{beamer}
\usepackage{lmodern}
\usepackage[T1]{fontenc}

\begin{document}
\begin{frame}{some frame}
\begin{itemize}[<+->]
\item there is foo
\item foo is good
\item and there is bar
\item bar is good too
\end{itemize}
\end{frame}
```

```
\begin{frame}<1-2>[label=lfoo]{frame on foo}
\begin{itemize}[<+->]
\item item a
\item item b \uncover<2>{\hyperlink{lfoo<3>}%
                            {\beamerskipbutton{Skip Details}}}
\item item c
\item item d
\end{itemize}
\end{frame}

\begin{frame}{details on foo item b}
\begin{itemize}[<+->]
\item item b.1
\item item b.2
\item item b.3
\item item b.4
\end{itemize}
\end{frame}

\begin{frame}
\frametitle{more details on foo item b}
\begin{itemize}[<+->]
\item item b.5
\item item b.6
\item item b.7
\item item b.8
\end{itemize}
\end{frame}

\againframe<3->{lfoo}

\begin{frame}{frame on bar}
\begin{itemize}[<+->]
\item item w
\item item x
\item item y
\item item z \uncover<4>{\hyperlink{lother}%
                            {\beamerskipbutton{Skip Details}}}
\end{itemize}
\end{frame}

\begin{frame}
\frametitle{details on bar item z}
\begin{itemize}[<+->]
\item item z.1
\item item z.2
\item item z.3
\item item z.4
\end{itemize}
```

```
\vspace{2ex}
\uncover<4>{\hyperlink{lother}{\beamerskipbutton{Skip further Details}}}
\end{frame}

\begin{frame}{more details on bar item z}
\begin{itemize}[<+->]
\item item z.5
\item item z.6
\item item z.7
\item item z.8
\end{itemize}
\end{frame}

\begin{frame}[label=lother]{another frame}
\begin{itemize}[<+->]
\item conclusion 1
\item conclusion 2
\item conclusion 3
\item conclusion 4
\end{itemize}
\end{frame}
\end{document}
```

4.4.2 With BeamerSubFrame

And now the same order of frames, but this time using the BeamerSubFrame package and with some comments.

```
\documentclass{beamer}
\usepackage{lmodern}
\usepackage[T1]{fontenc}
```

Here, the option `embed` has to be changed to `append`, to get the order as shown in Figure 2.

```
\usepackage[embed]{beamersubframe}
```

The two commands, as explained in subsection 4.3.

```
\newcommand<>{\linkdetail}[4]{%
    \ifappend{\uncover#5{\hyperlink{#3}{\beamergotobutton{#4}}}}%
             {\uncover#5{\hyperlink{#1}{\beamerskipbutton{#2}}}}
}
\newcommand<>{\returndetail}[2]{%
\ifappend{\uncover#3{\hyperlink{#1}{\beamerreturnbutton{#2}}}}%
         {\visible<0>{\beamerreturnbutton{#2}}}
}
```

The next command realises a link with the same destination (argument `#1`) for both modes, but different texts (`#2` for embed mode and `#3` for append mode.

```
\newcommand<>{\linkdifftext}[3]{%
    \ifappend{\uncover#4{\hyperlink{#1}{\beamergotobutton{#3}}}}%
             {\uncover#4{\hyperlink{#1}{\beamerskipbutton{#2}}}}
}
```

The template defined below is an example for the use of `\ifsubframe`. It puts the text "detail" in the lower left corner of subframes and "main part" in the lower left corner of normal frames. Careful: it replaces a sidebar, so this will not work with themes like Berkeley.

```
\defbeamertemplate*{sidebar left}{bsf test}[1][50]
{
  \vfill%
  \rlap{\hskip0.1cm\hbox{\color{fg!#1!bg}\tiny{\ifsubframe{detail}{main part}}}}%
  \vskip2pt%
}

\begin{document}
\begin{frame}{some frame}
\begin{itemize}[<+->]
\item there is foo
\item foo is good
\item and there is bar
\item bar is good too
\end{itemize}
\end{frame}

\begin{frame}<1-2>[label=lfoo]{frame on foo}
\begin{itemize}[<+->]
\item item a
```

Here, the link was changed.

```
\item item b \linkdetail<2>{lfoo<3>}{Skip Details}{ldfoob}{Goto Details}
\item item c
\item item d
\end{itemize}
\end{frame}
```

For the next two frames the environment was changed to `subframe`. And for the first, the label `ldfoob` was added.

```
\begin{subframe}[label=ldfoob]{details on foo item b}
\begin{itemize}[<+->]
\item item b.1
```

```
\item item b.2
\item item b.3
\item item b.4
\end{itemize}
\end{subframe}

\begin{subframe}
\frametitle{more details on foo item b}
\begin{itemize}[<+->]
\item item b.5
\item item b.6
\item item b.7
\item item b.8
\end{itemize}

\vspace{2ex}
```

The link to return from details was added. Because there was no link in the version without BeamerSubFrame the layout of the frame is now changed compared to that version. But it doesn't change from one mode to the other, because of the way `\returndetails` was definded.

```
\returndetail<4>{lfoo<3>}{Return from Details}
\end{subframe}

\againframe<3->{lfoo}

\begin{frame}{frame on bar}
\begin{itemize}[<+->]
\item item w
\item item x
\item item y
```

Again, the link was changed here.

```
\item item z \linkdetail<4>{lother}{Skip Details}{ldbarz}{Goto Details}
\end{itemize}
\end{frame}
```

For the next two frames the environment was changed to `subframe` also. And for the first, the label `ldbarz` was added.

```
\begin{subframe}[label=ldbarz]
\frametitle{details on bar item z}
\begin{itemize}[<+->]
\item item z.1
\item item z.2
\item item z.3
```

```
\item item z.4
\end{itemize}

\vspace{2ex}
```

This link was changed, so the text differs between modes. But it would have been possible here, to leave the link unchanged.

```
\linkdifftext<4>{lother}{Skip further Details}{Return from Details}
\end{subframe}

\begin{subframe}{more details on bar item z}
\begin{itemize}[<+->]
\item item z.5
\item item z.6
\item item z.7
\item item z.8
\end{itemize}

\vspace{2ex}
```

Again, the link was added here, resulting also in a change off layout compared to the first version.

```
\returndetail<4>{lother}{Return from Details}
\end{subframe}

\begin{frame}[label=lother]{another frame}
\begin{itemize}[<+->]
\item conclusion 1
\item conclusion 2
\item conclusion 3
\item conclusion 4
\end{itemize}
\end{frame}
```

And finally, the `\appendsubframes` command was inserted here.

```
\appendsubframes
\end{document}
```

The source of the second version is available as example file, which can be generated from `beamersubframe.dtx`.

5 Restrictions

Since this is an early version of the BeamerSubFrame package, there are some restrictions.

- The package was only tested with Beamer mode `beamer`. Using other modes may lead to strange errors.
- Supporting material, like notes, is not tested yet.
- Lectures are not tested yet.
- There is no support for subsubsections (not even tested). And it is not planned at all.
- There is no `\subframe` command.
- There is no `\againsubframe` command.

6 Testing

For testing the BeamerSubFrame package the following themes were used:

- Dresden
- Darmstadt
- Berkeley
- Montpellier
- Malmoe
- AnnArbor

The themes were used without options. They where choosen to cover all types of navigation possible with the themes delivered with Beamer. If I missed something and this doesn't work, please let me know. If there are other things not working (e.g. some inserts), please let me know too. In both cases, a file for testing would be nice.

7 ToDo

Although the BeamerSubFrame package is already useful, there are a lot of things left to do:

testing Beamer has a lot of features, an many of them are still not tested.

Beamer modes I have to admit, I didn't even test them.

notes The same here.

lectures And again, not tested.

macros There are commands, which would be nice to have, like `\subframe`, `\againsubframe`, and `\subnote` (to get notes between frames moved too)

levels With multiple levels of subframes, it would be possible, to have details on details (on details on ...). And then, one could embed the first n levels of details, and append the others.

The list is not complete. If someone has an item to add, please let me know.

8 The Code

8.1 The Usual

First the usual things.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{beamersubframe}[\filedate\space
    v\fileversion\space reordering beamer frames]
```

8.2 Check the Class

Since the BeamerSubFrame package only works with the Beamer class and article mode is not supported yet, check for the Beamer class. If it is not loaded, throw an error message and stop loading the package.

```
\@ifclassloaded{beamer}{}{%
  \PackageError{beamersubframe}{%
    The package works only with the beamer class,\MessageBreak
    therefore it is not loaded.
  }{%
    The package is not loaded, because it needs the\MessageBreak
    beamer class. Continuing may lead to additional\MessageBreak
    errors because of undefined commands.
  }
  \endinput
}
```

8.3 "Variables"

`\if@bsf@append` First there is the main flag to distinguish between append mode and embed mode.

```
\newif\if@bsf@append
```

`\if@bsf@miniframes` This flag is true by default. It is set to false by the option `nominiframes`.

```
\newif\if@bsf@miniframes
\@bsf@miniframestrue
```

`\if@bsf@subframe` This flag is set to true for subframes in both modes. For normal frames it is false.

```
\newif\if@bsf@subframe
\@bsf@subframefalse
```

`\if@bsf@nosfnum` This flag is used for checking the `.sfp` file.

```
\newif\if@bsf@nosfnum
```

`\if@bsf@firstline` This flag is needed to write `\begin{frame}` in front of the arguments of a `subframe` environment to the `.sfr` file in append mode.

```
\newif\if@bsf@firstline
```

`\if@bsf@firstpart` This flag is needed by `\bsfrestorepart` to initialize some macros used as variables instead of writing the `\bsf@partpages` entry to the `.nav` file.

```
\newif\if@bsf@firstpart
```

`\if@bsf@firstsection` This flag is needed by `\bsfrestoresection` to initialize some macros used as variables instead of writing the `\bsf@sectionpages` entry to the `.nav` file.

```
\newif\if@bsf@firstsection
```

`\if@bsf@firstsubsection` This flag is needed by `\bsfrestoresubsection` to initialize some macros used as variables instead of writing the `\bsf@subsectionpages` entry to the `.nav` file.

```
\newif\if@bsf@firstsubsection
```

`\if@bsf@nosubsection` This flag is used by `\bsfrestoresection` and `\bsfrestoresubsection` to determine, if a section contains any subsections.

```
\newif\if@bsf@nosubsection
```

`\if@bsf@prevnosubsection` This flag contains the final state of `\if@bsf@nosubsection` for the previous section.

```
\newif\if@bsf@prevnosubsection
```

`\bsf@frame@param` This token register is used to store the first three arguments of the `subframe` environment in embed mode, after adding the option `environmet=subframe`. The contents is passed to the `frame` environmet as its first three arguments.

```
\newtoks\bsf@frame@param
```

`\bsf@sfrout` The stream used to write the `.sfr` file. It holds the contents of all subframes. The file is only written and used in append mode.

```
\newwrite\bsf@sfrout
```

8.4 Options and Packages

`embed` The options are rather simple. The option `embed` just sets `\if@bsf@append` to false.

```
\DeclareOption{embed}{\@bsf@appendfalse}
```

`append` And `append` sets `\if@bsf@append` to true.

```
\DeclareOption{append}{\@bsf@appendtrue}
```

`nominiframes` The option `nominiframes` sets `\if@bsf@miniframes` to false.

```
\DeclareOption{nominiframes}{\@bsf@miniframesfalse}
```

The default is set and the options are processed.

```
\ExecuteOptions{embed}
\ProcessOptions*\relax
```

And the verbatim package is loaded.

```
\RequirePackage{verbatim}
```

8.5 Navigation

8.5.1 Sidebar and Headline

To get the highlighting of sections and subsections on appened subframes as if the frames were embedded, some counters must be restored to the values, they had at the point, the subframe appeared in the source. Additionally, for the headline some inserts must be restored.

To achieve this, the `subframe` environment writes a macro with all necessary arguments to the `.sfr` file. This is done for each subframe, before its contents, embedded in a `frame` environmet, is written to the file (see subsubsection 8.5.5).

`\bsfrestore` The macro is called `\bsfrestore`. It is executed, when the `.sfr` file is loaded. It restores the counters for part (`#1`), section (`#2`), and aubsection (`#3`). It also restores the counter subsectionslide (`#4`), which is needed for miniframes.

Then `\insertsectionhead` and `\insertsubsectionhead` are restored to their meanings at the point, the subframe appeard in the source. These meanings were stored to the two macros `\bsf@sectionhead`⟨*part*⟩`.`⟨*section*⟩ and `\bsf@subsectionhead`⟨*part*⟩`.`⟨*section*⟩`.`⟨*subsection*⟩ by the `subframe` environment (see subsubsection 8.5.5).

After that, the inserts for the part, section, and subsection numbers are restored.

The macro is also used by the `lastframe` environment (see subsubsection 8.5.6).

```
\newcommand{\bsfrestore}[4]{%
  \setcounter{part}{#1}%
  \setcounter{section}{#2}%
  \setcounter{subsection}{#3}%
  \setcounter{subsectionslide}{#4}%
  \expandafter\let\expandafter\insertsectionhead
    \csname bsf@sectionhead\the\c@part.\the\c@section\endcsname
  \expandafter\let\expandafter\insertsubsectionhead
    \csname bsf@subsectionhead\the\c@part.\the\c@section.\the\c@subsection\endcsname
  \def\insertpartheadnumber{#1}%
  \def\insertsectionheadnumber{#2}%
  \def\insertsubsectionheadnumber{#3}%
}
```

8.5.2 Miniframes

To typset miniframes, Beamer writes a `\slideentry` with the necessary arguments to the `.nav` file. Because miniframes should appear, as if the subframes are embedded, the `subframe` environment has to write such an entry at the point, the subframe appears in the source. But instead of a `\slideentry`, BeamerSubFrame writes a `\subslideentry` (see subsection 2.10).

Argument `#4` of `\slideentry` contains the start page and the end page of the frame. Unfortunately, for subframes these numbers are unknown, until the `.sfr` file was loaded by `\appendsubframes`.

To get around this, for each subframe two macros are used. Their names are `\bsf@substartpage`⟨*part*⟩`.`⟨*section*⟩`.`⟨*subsection*⟩`.`⟨*subsectionslide*⟩ and `\bsf@subendpage`⟨*part*⟩`.`⟨*section*⟩`.`⟨*subsection*⟩`.`⟨*subsectionslide*⟩. Here ⟨*part*⟩, ⟨*section*⟩ and so on are the numbers.

Now at the point where the subframe appears in the souce, the `\subslideentry` is written. Instead of the unknown frame numbers, it contains the sequence `{\bsf@usenum{bsf@substartpage...}/\bsf@usenum{bsf@subendpage...}}` as argument `#4`. After a subframe was typeset by loading the `.sfr` file, the necessary information is written to the `.sfp` file. In the next TEX run, when loading the `.sfp` file at the beginning of the document (see subsubsection 8.5.8), these macros are all defined.

`\bsf@subnum` The first macro involved is just a shortcut to get the numbers.

```
\def\bsf@subnum{%
  \the\c@part.\the\c@section.\the\c@subsection.\the\c@subsectionslide
}
```

Beamer uses the macro `\beamer@writeslidentry` to write the `\slideentry` and the `\beamer@framepages` entry (used for the navigation bar) to the `.nav` file. For subframes this is split into two macros.

`eamer@writeslidentry@miniframes` The `\subslideentry` is written by `\beamer@writeslidentry@miniframes`. The macro is called directly by the `subframe` environment (see subsubsection 8.5.5).

```
\def\beamer@writeslidentry@miniframes{%
  \addtocontents{nav}%
    {\protect\headcommand{%
      \protect\subslideentry{\the\c@section}{\the\c@subsection}%
        {\the\c@subsectionslide}%
        {\protect\bsf@usenum{bsf@substartpage\bsf@subnum}/%
         \protect\bsf@usenum{bsf@subendpage\bsf@subnum}}%
        {\lastsubsection}{\the\c@part}}}%
}
```

`\beamer@writeslidentry@navbar` The `\beamer@framepages` entry is written by `\beamer@writeslidentry@navbar`. And it also writes the `\bsfsubframepages` entry to the `.sfp` file.

This macro is used to replace `\beamer@writeslidentry`, before the `.sfr` file is loaded. This is done in `\appendsubframes` (see subsubsection 8.5.7).

```
\def\beamer@writeslidentry@navbar{%
  \expandafter\beamer@ifempty\expandafter{\beamer@framestartpage}{}% does not happen normally
  {%else
    \addtocontents{nav}%
      {\protect\headcommand{%
        \protect\beamer@framepages{\beamer@framestartpage}{\beamer@frameendpage}}}%
    \addtocontents{sfp}{%
      \protect\bsfsubframepages{\the\c@part}{\the\c@section}{\the\c@subsection}%
        {\the\c@subsectionslide}{\beamer@framestartpage}{\beamer@frameendpage}}%
    \clearpage\beamer@notesactions%
  }%
}
```

\subslideentry The \subslideentry is initialized in a way that it does the same as the \slideentry. Because \slideentry is refedined by some themes, \let can not be used here.

```
\def\subslideentry{\slideentry}
```

\bsfsubframepages The .sfp file contains one \bsfsubframepages entry for each subframe. The arguments are the part number (#1), the section number (#2), the subsection number (#3), the subsectionslide number (#4), and the first (#5) and the last page of the subframe (#6).

When the .sfp file is loaded, each \bsfsubframepages defines two macros \bsf@substartpage... and \bsf@subendpage... used by the \subslideentry.

```
\def\bsfsubframepages#1#2#3#4#5#6{%
  \expandafter\def\csname bsf@substartpage#1.#2.#3.#4\endcsname{#5}%
  \expandafter\def\csname bsf@subendpage#1.#2.#3.#4\endcsname{#6}%
}
```

\bsf@usenum The macro \bsf@usenum, used in the \subslideentry, sets a default in case the \bsf@substartpage... and \bsf@subendpage... macros are not defined due to a missing, corrupted or incomplete .sfp file. Without this, there could be errors which would persist on every subsequent run of TEX.

```
\def\bsf@usenum#1{%
  \@ifundefined{#1}{1}{\csname #1\endcsname}%
}
```

\bsf@checksfp In order to warn the user about a missing, corrupted or incomplete .sfp file, the file is checked. This is done indirectly by checking, if all \bsf@substartpage... and \bsf@subendpage... macros in the current .nav file are defined.

For this all commands in the .nav file are executed using Beamers \dohead command. Because some of the commands would typeset something, they are disabled and everything is done in a group to keep the changes local.

The macro \bsf@usenum is also redefined to set the flag \if@bsf@nosfnum to true in case one of the checked macros is undefined.

```
\def\bsf@checksfp{%
  \begingroup
    \@bsf@nosfnumfalse
    \def\bsf@usenum##1{%
      \@ifundefined{##1}{\@bsf@nosfnumtrue}{}}
    \def\slideentry##1##2##3##4##5##6{}%
    \def\partentry##1##2{}%
    \def\sectionentry##1##2##3##4##5{}%
    \def\beamer@subsectionentry##1##2##3##4##5{}%
    \def\subslideentry##1##2##3##4##5##6{%
      \bsf@checksfnum(##4)}%
    \dohead
    \if@bsf@nosfnum
      \PackageWarningNoLine{Beamer SubFrame}{%
        Missing, incomplete or corrupted file\MessageBreak
        \jobname.sfp! Links for miniframes of\MessageBreak
```

```
96         subframes may be wrong. Please run TeX\MessageBreak
97         again}
98     \fi
99   \endgroup
100 }
```

`\bsf@checksfnum` The macro `\bsf@checksfnum` is just used by `\bsf@checksfp` to get rid of the slash in argument `#4` of the `\subslideentry`.

```
101 \def\bsf@checksfnum(#1/#2){#1#2}
```

8.5.3 Navigation Bar

Getting the navigation bar right takes some effort. First lets take a look, what Beamer does to get the necessary information for the navigation bar.

How Beamer does it
Beamer writes several different entries to the `.nav` file.

`\beamer@framepages` with the first and the last page of the frame as arguments. It is written for every frame at its end.

`\beamer@subsectionpages` with the first and the last page of the subsection as arguments. It is written by the `\subsection` command for the previous subsection, as well as by `\section` and `\part` and the end of the document for the last subsection.

`\beamer@sectionpages` with the first and the last page of the section as arguments. It is written by the `\section` command for the previous section, as well as by `\part` and the end of the document for the last section.

`\beamer@partpages` with the first and the last page of the part as arguments. It is written by the `\part` command for the previous part and the end of the document for the last part.

`\beamer@documentpages` with the last page of the documnet as argument. It is written at the end of the document.

The macros in the `.nav` file are executed once for every page (i.e. slide). By this the entry `\beamer@framepages` sets the macro `\beamer@startpageofframe` to its first argument and `\beamer@endpageofframe` to its second argument, but only if the current page is $\geq$ than the first argument and $\leq$ than the second one. Similar macros are set the same way by the other entries.

The frame icon is then typeset in a way, that

- the left arrow points to page `\beamer@startpageofframe` -1,
- the left part of the frame symbol points to page `\beamer@startpageofframe`,
- the right part of the frame symbol points to page `\beamer@endpageofframe`,

- and the right arrow points to page `\beamer@endpageofframe` +1.

The arrows are limited 1 or `\beamer@endpageofdocument` respectively. The icons for subsections and sections are typeset in a similar way.

The right part of the presentation symbol or (if there is an appendix) of the appendix symbol will point to page `\beamer@endpageofdocument`.

How BeamerSubFrame does it

The frame icon should point to the previous or next frame or subframe in the PDF file. This is simply achieved by writing the `\beamer@framepages` entry with `\beamer@writeslidentry@navbar` (see subsubsection 8.5.2) for every subframe.

The section icon should point to the first and last page of the section the subframe belongs to in the main part. For this, the first page of the first and the last page of the last subframe of a section is needed also. The information is stored in two steps.

1. A macro `\bsfrestoresection` is written to the `.sfr` file for each section. Its arguments are the number of the previous section and the first page of the section just started.

2. When loading the `.sfr` file, the first `\bsfrestoresection` entry initializes some makros (used as variables). All subsequent entries write a `\bsf@sectionpages` entry to the `.nav` file, which has the necessary pages as arguments.

For parts and subsections corresponding entries are written to the files. And for the last part, section, and subsection `\bsf@...pages` entries are written after appending the subframes.

The `\beamer@...pages` entries written by Beamer at the end of the document would mix up the links of the navigation bar. It is not possible to prevent writing these entries. Instead they are disabled after appending the subframes. Since they would stay disabled for all pages after the first, they are reenabled at the beginning of the `.nav` file. Also, these entries have to be written before appending the subframes, in order to make the navigation bar work correctly for the last section and subsection in the main part.

Since this is also done for the `\beamer@documentpages` entry, the right arrows of the section and subsection icons are limited to the last page of the main part, i.e. the page before the first subframe. The same applies to the right part of the presentation or appendix symbol.

Unfortunately, this would apply to the right arrows of the slide icon and the frame icon too. Therefore the package writes a `\bsf@documentpages` entry to the `.nav` file after appending the subframes. Additionally, two macros of Beamer are redefined to use this information, instead of the information of the `\beamer@documentpages` entry.

The Macros

Lets start with the two redefined macros. The originals are defined in beamerbasenavigation.sty.

\hyperlinkslidenext The new definition of \hyperlinkslidenext uses \bsf@nextpage instead of \beamer@nextpage.

```
\def\hyperlinkslidenext{%
  \bsf@nextpage\c@page%
  \hyperlink{Navigation\the\beamer@tempcount}}
```

\hyperlinkframestartnext The new definition of \hyperlinkframestartnext also uses \bsf@nextpage instead of \beamer@nextpage.

```
\def\hyperlinkframestartnext{%
  \bsf@nextpage\beamer@endpageofframe%
  \hyperlink{Navigation\the\beamer@tempcount}}
```

\bsf@nextpage The macro \bsf@nextpage works like \beamer@nextpage (from beamerbasenavigation.sty), but it uses \bsf@endpageofdocument as the last page for links, instead of \beamer@endpageofdocument.

```
\def\bsf@nextpage#1{%
  \beamer@tempcount=#1%
  \advance\beamer@tempcount by1\relax%
  \ifnum\beamer@tempcount>\bsf@endpageofdocument%
  \beamer@tempcount=\bsf@endpageofdocument%
  \fi}
```

\bsf@documentpages The \bsf@endpageofdocument is defined by the \bsf@documentpages entry of the .nav file.

```
\def\bsf@documentpages#1{\def\bsf@endpageofdocument{#1}}
```

\bsf@endpageofdocument And \bsf@endpageofdocument is initialized to reflect the original.

```
\def\bsf@endpageofdocument{\beamer@endpageofdocument}
```

To disable and reenable the \beamer@...pages entries, first

\beamer@partpages@orig the macro \beamer@partpages,

```
\let\beamer@partpages@orig\beamer@partpages
```

\beamer@subsectionpages@orig the macro \beamer@subsectionpages,

```
\let\beamer@subsectionpages@orig\beamer@subsectionpages
```

\beamer@sectionpages@orig the macro \beamer@sectionpages,

```
\let\beamer@sectionpages@orig\beamer@sectionpages
```

\beamer@documentpages@orig and the macro \beamer@documentpages are saved.

```
\let\beamer@documentpages@orig\beamer@documentpages
```

\bsf@enablenaventries The command \bsf@enablenaventries restores the original meanings of the \beamer@...pages entries. It is written to the .nav file at the beginning of the document (see subsubsection 8.5.8).

```
\def\bsf@enablenaventries{%
```

```
  \let\beamer@partpages\beamer@partpages@orig
  \let\beamer@subsectionpages\beamer@subsectionpages@orig
  \let\beamer@sectionpages\beamer@sectionpages@orig
  \let\beamer@documentpages\beamer@documentpages@orig
}
```

`\bsf@disablenaventries` And `\bsf@disablenaventries` disables the entries by letting them gobble their arguments. It is written to the `.nav` file after appending the subframes (see subsubsection 8.5.7).

```
\def\bsf@disablenaventries{%
  \let\beamer@partpages\@gobbletwo
  \let\beamer@subsectionpages\@gobbletwo
  \let\beamer@sectionpages\@gobbletwo
  \let\beamer@documentpages\@gobble
}
```

In order to write the `\bsfrestore...` entries to the `.sfr` file, the sectioning commands must be extended. First the originals for

`\part@orig` the macro `\part`,

```
\let\part@orig\part
```

`\section@orig` the macro `\section`,

```
\let\section@orig\section
```

`\subsection@orig` and the macro `\subsection` are saved.

```
\let\subsection@orig\subsection
```

The sectioning macros are only redefined in append mode.

```
\if@bsf@append
```

`\part` The new version of `\part` writes the `\bsfrestorepart` entry to the `.sfr` file. The original version is called at the end, so its arguments are of no concern here. But because of that, `\c@part` still contains the number of the previous part. Here, `\c@page` already contains the number of the first page of the new part.

```
  \def\part{%
    \immediate\write\bsf@sfrout{\string\bsfrestorepart{\the\c@part}%
        {\the\c@page}}%
    \part@orig
  }
```

`\section` The new version of `\section` writes the `\bsfrestoresection` entry to the `.sfr` file. Besides that, it works like the new `\part` command.

```
  \def\section{%
    \immediate\write\bsf@sfrout{\string\bsfrestoresection{\the\c@section}%
        {\the\c@page}}%
    \section@orig
  }
```

\subsection And the new version of \subsection writes the \bsfrestoresubsection entry to the .sfr file. It also works like the new \part command.

```
\def\subsection{%
  \immediate\write\bsf@sfrout{\string\bsfrestoresubsection{\the\c@subsection}%
      {\the\c@page}}%
  \subsection@orig
}
\fi
```

The \bsfrestore... entries are executed, when loading the .sfr file. Basically, they write the \bsf@... pages entries to the .nav file. But the first time, they just initialize some macros used as variables, because at this point only the first page of the first subframe of a part, section or subsection and the first page of the part, section or subsection itself are known.

\bsfrestorepart On the first call, \bsfrestorepart stores the the first page of the new part (#2), the number of the previous part (#1), and the number of the first page of the first subframe belonging to the new part (c@page).

```
\def\bsfrestorepart#1#2{%
  \if@bsf@firstpart
    \@bsf@firstpartfalse
    \def\bsf@partstartpage{#2}%
    \def\bsf@prevpart{#1}%
    \edef\bsf@partfirstsubframepage{\the\c@page}%
  \else
```

On all subsequent calls, if there is a new part, the last page of the last subframe belonging to the previous part (in \@tempcnta) and the last page of the previous part (in \@tempcntb) are calculated.

```
    \@tempcnta=\bsf@prevpart\relax
    \ifnum#1>\@tempcnta
      \@tempcnta\c@page\advance\@tempcnta -1\relax
      \@tempcntb=#2\relax\advance\@tempcntb -1\relax
```

Then the \bsf@partpages entry is written to the .nav file. Here \addtocontents can not be used, because it doesn't work, if \bsfrestorepart appears after the last subframe.

```
      \if@filesw
        \immediate\write\@auxout{\string\@writefile{nav}%
          {\noexpand\headcommand{%
            \noexpand\bsf@partpages{\bsf@partfirstsubframepage}%
              {\the\@tempcnta}{\bsf@partstartpage}{\the\@tempcntb}}}}%
      \fi
```

After that, the "variables" are reinitialized.

```
      \def\bsf@partstartpage{#2}%
      \def\bsf@prevpart{#1}%
      \edef\bsf@partfirstsubframepage{\the\c@page}%
    \fi
  \fi
```

```
}
```

Two of the "variables" must be initialized, in order to make the package work for presentations without sectioning commands.

```
\def\bsf@partstartpage{1}%
\def\bsf@prevpart{0}%
```

`\bsfrestoresection` The macros `\bsfrestoresection` and `\bsfrestoresubsection` work the same way as `\bsfrestorepart`. But additionally, they must keep track of sections without subsections. For this, the flags `\if@bsf@nosubsection` and `\if@bsf@prevnosubsection` and the macro `\bsf@prevsectionstartpage` are used.

```
\def\bsfrestoresection#1#2{%
  \if@bsf@firstsection
    \@bsf@firstsectionfalse
    \def\bsf@prevsectionstartpage{#2}%
    \@bsf@nosubsectiontrue
    \@bsf@prevnosubsectionfalse
    \def\bsf@sectionstartpage{#2}%
    \def\bsf@prevsection{#1}%
    \edef\bsf@sectionfirstsubframepage{\the\c@page}%
  \else
    \@tempcnta=\bsf@prevsection\relax
    \ifnum#1>\@tempcnta
```

When a new section starts, the start page and the `\if@bsf@nosubsection` flag of the previous section are stored.

```
      \edef\bsf@prevsectionstartpage{\bsf@sectionstartpage}%
      \if@bsf@nosubsection
        \@bsf@prevnosubsectiontrue
      \else
        \@bsf@prevnosubsectionfalse
      \fi
```

The flag `\if@bsf@nosubsection` is set to false, because the new section has no subsections yet.

```
      \@bsf@nosubsectiontrue
```

And the number of the previous subsection is set to -1, because the subsection counter will be reset to 0 on the start of each section.

```
      \def\bsf@prevsubsection{-1}%
      \@tempcnta\c@page\advance\@tempcnta -1\relax
      \@tempcntb=#2\relax\advance\@tempcntb -1\relax
      \if@filesw
        \immediate\write\@auxout{\string\@writefile{nav}%
          {\noexpand\headcommand{%
            \noexpand\bsf@sectionpages{\bsf@sectionfirstsubframepage}%
              {\the\@tempcnta}{\bsf@sectionstartpage}{\the\@tempcntb}}}}%
      \fi
      \def\bsf@sectionstartpage{#2}%
```

```
      \def\bsf@prevsection{#1}%
      \edef\bsf@sectionfirstsubframepage{\the\c@page}%
    \fi
  \fi
}
\def\bsf@sectionstartpage{1}%
\def\bsf@prevsection{0}%
```

`\bsfrestoresubsection` The macro `\bsfrestoresubsection` uses the flags `\if@bsf@nosubsection` and `\if@bsf@prevnosubsection` and the macro `\bsf@prevsectionstartpage` to set the correct range of pages.

```
\def\bsfrestoresubsection#1#2{%
  \if@bsf@firstsubsection
    \@bsf@firstsubsectionfalse
    \@bsf@prevnosubsectionfalse
    \def\bsf@subsectionstartpage{#2}%
    \def\bsf@prevsubsection{#1}%
    \edef\bsf@subsectionfirstsubframepage{\the\c@page}%
  \else
    \@tempcnta=\bsf@prevsubsection\relax
    \ifnum#1>\@tempcnta
      \@tempcnta\c@page\advance\@tempcnta -1\relax
```

Since the `\bsf@...pages` entries are written (indirectly) at the point of the next part, section or subsection, the entry for let's say subsection 1.3 would be written at the point of subsection 3.1, if section 2 has no subsections. So if the previous section had no subsection, its start page is used to calculate the last page of the last subsection instead of the start page of this subsection.

```
      \if@bsf@prevnosubsection
        \@tempcntb=\bsf@prevsectionstartpage\relax
      \else
        \@tempcntb=#2\relax
      \fi
      \advance\@tempcntb -1\relax
```

Because `\if@bsf@prevnosubsection` can only be used at the point of the first subsection of a new section, it is set to false.

```
      \@bsf@prevnosubsectionfalse
      \@bsf@nosubsectionfalse
      \if@filesw
        \immediate\write\@auxout{\string\@writefile{nav}%
          {\noexpand\headcommand{%
            \noexpand\bsf@subsectionpages{\bsf@subsectionfirstsubframepage}%
              {\the\@tempcnta}{\bsf@subsectionstartpage}{\the\@tempcntb}}}}%
      \fi
      \def\bsf@subsectionstartpage{#2}%
      \def\bsf@prevsubsection{#1}%
      \edef\bsf@subsectionfirstsubframepage{\the\c@page}%
    \fi
  \fi
```

```
}
\def\bsf@subsectionstartpage{1}%
\def\bsf@prevsubsection{0}%
```

`\bsf@partpages` The `\bsf@partpages` macro is similar to the `\beamer@partpages` macro (from beamerbasenavigation.sty). But to decide if the first and the last page of a part are stored in `\beamer@startpageofpart` and `\beamer@endpageofpart`, it uses the first page of the first subframe (`#1`) and the last page of the last subframe (`#2`) belonging to the part.

There may be parts with no subframes. In this case, `#1` is greater then `#2` and the first (`#3`) and last page (`#4`) of the part are not stored.

```
\def\bsf@partpages#1#2#3#4{%
  \ifnum\c@page<#1%
  \else%
    \ifnum\c@page>#2%
    \else%
      \gdef\beamer@startpageofpart{#3}%
      \gdef\beamer@endpageofpart{#4}%
    \fi%
  \fi%
}
```

`\bsf@sectionpages` The macro `\bsf@sectionpages` works the same way as `\bsf@partpages`.

```
\def\bsf@sectionpages#1#2#3#4{%
  \ifnum\c@page<#1%
  \else%
    \ifnum\c@page>#2%
    \else%
      \gdef\beamer@startpageofsection{#3}%
      \gdef\beamer@endpageofsection{#4}%
    \fi%
  \fi%
}
```

`\bsf@subsectionpages` And `\bsf@sectionpages` works the same way as `\bsf@partpages` too.

```
\def\bsf@subsectionpages#1#2#3#4{%
  \ifnum\c@page<#1%
  \else%
    \ifnum\c@page>#2%
    \else%
      \gdef\beamer@startpageofsubsection{#3}%
      \gdef\beamer@endpageofsubsection{#4}%
    \fi%
  \fi%
}
```

8.5.4 Inserts

Beamers `\inserttotalframenumber` is changed by simply writing its definition a second time to the `.nav` file, thus overwriting Beamers original definition. This is done at the end of the document (see subsubsection 8.5.8). The necessary frame number is stored in `\bsf@totalframenumber` before appending the subframes (see subsubsection 8.5.7).

`\inserttotalframenumberwithsub` The new `\inserttotalframenumberwithsub` is written at the end of the documnet (see subsubsection 8.5.8), regardless of the mode of BeamerSubFrame, so it's always defined. But to work on the first TEX run, it must be initialized.

```
\def\inserttotalframenumberwithsub{\inserttotalframenumber}
```

8.5.5 The `subframe` environment

`subframe` The `subframe` environment is defined differently for the two modes. In append mode it is basically a verbatim environment, which writes its contents to the `.sfr` file. Here, the verbatim package is used.

Before writing the contents, it writes the `\bsfrestore` entry for the subframe. The latter contains the value of the `subsectionslide` counter as fourth argument, which has to be corrected, if miniframes for subframes should not appear. The counter needs to be incremented, if miniframes should appear for subframes.

The `\bsfrestore` entry also has to restore the section and subsection heads. Since they might contain macros and not just the text itself, `\insertsectionhead` and `\insertsubsectionhead` can not be used directly. Their current meaning is stored in macros, which are later used to restore the heads.

The first line of the `subframe` environment, which contains the arguments, is treated differently. It is preceded with `\begin{frame}`.

At the end of the environment, `\end{frame}` is written to the `.sfr` file. And the `\subslideentry` is written with `\beamer@writeslidentry@miniframes`, but only if miniframes should appear for subframes.

```
\if@bsf@append
  \newenvironment{subframe}{%
    \@tempcnta\c@subsectionslide
    \if@bsf@miniframes\else \advance\@tempcnta by -1\fi
    \expandafter\global\expandafter\let
      \csname bsf@sectionhead\the\c@part.\the\c@section
        \endcsname\insertsectionhead
    \expandafter\global\expandafter\let
      \csname bsf@subsectionhead\the\c@part.\the\c@section.\the\c@subsection
        \endcsname\insertsubsectionhead
    \immediate\write\bsf@sfrout{\string\bsfrestore{\the\c@part}{\the\c@section}%
       {\the\c@subsection}{\the\@tempcnta}}%
    \if@bsf@miniframes \addtocounter{subsectionslide}{1}\fi
    \@bsf@firstlinetrue
    \let\do\@makeother\dospecials\catcode`\^^M\active
    \def\verbatim@processline{%
      \if@bsf@firstline
```

```
        \immediate\write\bsf@sfrout{\string\begin{frame}\the\verbatim@line}%
        \@bsf@firstlinefalse
      \else
        \immediate\write\bsf@sfrout{\the\verbatim@line}%
      \fi
    }%
    \verbatim@}{\immediate\write\bsf@sfrout{\string\end{frame}^^J}%
    \if@bsf@miniframes \beamer@writeslidentry@miniframes \fi
  }
\else
```

In embed mode, the `subframe` environment is basically a `frame` environment. Before calling it, `\if@bsf@subframe` is set to true. And after the `frame` environment, `\if@bsf@subframe` is set to false again. This is needed for the `\ifsubframe` command.

The main problem here is adding `environment=subframe` to the options. For this, `\bsf@frame` is called with the name of the environment, the optional overlay specification (`#2`), and the first optional argument (`#1`).

```
  \newenvironment<>{subframe}[1][]{%
    \@bsf@subframetrue
    \bsf@frame{subframe}{#2}{#1}}{\end{frame}\@bsf@subframefalse}
\fi
```

`\bsf@frame` The macro `\bsf@frame` initializes the token register `\bsf@frame@param`, sets a default for the second optional argument and then calls `\bsf@@frame`.

```
\def\bsf@frame#1#2#3{\bsf@frame@param={}%
  \@ifnextchar[{\bsf@@frame{#1}{#2}{#3}}{\bsf@@frame{#1}{#2}{#3}[]}%
}
```

`\bsf@@frame` The macro `\bsf@@frame` is called with the name of the new frame environment (`#1`), the overlay specification (`#2`), and the first (`#3`) and the second (`#4`) optional argument. It stores the arguments in`\bsf@frame@param`, thereby adding `environment=`⟨*name of the new frame environment*⟩ at the appropriate place.

```
\def\bsf@@frame#1#2#3[#4]{%
  \def\@tempa{#3}
```

If `#3` is empty, then there are no optional arguments, i.e. no default overlay specification and no options. In this case, `#4` is empty too.

```
  \ifx\@tempa\@empty
    \bsf@frame@param={#2[environment=#1]}%
  \else
    \def\@tempb{#4}
```

If only `#4` is empty, then there is one optional argument. This case is handled by `\bsf@@@frame`.

```
    \ifx\@tempb\@empty
      \bsf@@@frame{#1}{#2}#3\@@end
```

Otherwhise, both optional arguments are present.

```
    \else
```

```
      \bsf@frame@param={#2[#3][environment=#1,#4]}%
    \fi
  \fi
```

At the end, `\bsf@frame@` is called with the token register `\bsf@frame@param` already expanded.

```
  \expandafter\bsf@frame@\the\bsf@frame@param\@@end
}
```

`\bsf@@@frame` The macro `\bsf@@@frame` checks, if the only optional argument is a default overlay specification or the list of options. Here `#1` is the name of the new frame environment, `#2` is the overlay specification, `#3` is the first character of the optional argument and `#4` contains all subsequent characters.

If `#3` is '`<`', then the optional argument is a default overlay specification. Otherwhise, it is the list of options.

```
\def\bsf@@@frame#1#2#3#4\@@end{%
  \def\@tempa{#3}\def\@tempb{<}%
  \ifx\@tempa\@tempb\relax
    \bsf@frame@param={#2[#3#4][environment=#1]}%
  \else
    \bsf@frame@param={#2[environment=#1,#3#4]}%
  \fi
}
```

`\bsf@frame@` The macro `\bsf@frame@` calls `\begin{frame}` with its argument, which contains the overlay specification and the default overlay specification if present, and the options, containing at least `environment=...`. This includes angle brackets and brackets.

```
\def\bsf@frame@#1\@@end{\begin{frame}#1}
```

8.5.6 The `lastframe` environment

`lastframe` The `lastframe` environment also uses `\bsf@frame` to add `environment=lastframe` to the options before calling the `frame` environment.

Before that, it uses `\bsfrestore` to reset all sectioning counters. After this, the counters are in the same state, as they would be for a title frame before all sectioning commands. By this, navigation in the sidebar or headline will appear the same way as on the title page.

```
\newenvironment<>{lastframe}[1][]{%
  \bsfrestore{0}{0}{0}{0}%
  \bsf@frame{lastframe}{#2}{#1}}{\end{frame}}
```

8.5.7 Appending Subframes

`\appendsubframes` The macro `\appendsubframes` does nothing in embed mode. In append mode it first writes `\endinput` to the `.sfr` file and initializes some macros for navigation. The page number of the last frame of the main part is stored in `\bsf@pagebeforesub`, to be used after loading the `.sfr` file.

```
\newcommand{\appendsubframes}{%
  \if@bsf@append
    \immediate\write\bsf@sfrout{\string\endinput}%
    \edef\bsf@partfirstsubframepage{\the\c@page}%
    \edef\bsf@sectionfirstsubframepage{\the\c@page}%
    \edef\bsf@subsectionfirstsubframepage{\the\c@page}%
    \@tempcnta\c@page\advance\@tempcnta -1\relax
    \edef\bsf@pagebeforesub{\the\@tempcnta}%
```

For the navigation in the main part, the `\beamer@...pages` entries are written to the `.nav` file.

```
    \addtocontents{nav}{\protect\headcommand{%
        \protect\beamer@partpages{\the\beamer@partstartpage}{\bsf@pagebeforesub}}}%
    \addtocontents{nav}{\protect\headcommand{%
        \protect\beamer@subsectionpages{\the\beamer@subsectionstartpage}{\bsf@pagebeforesub}}}%
    \addtocontents{nav}{\protect\headcommand{%
        \protect\beamer@sectionpages{\the\beamer@sectionstartpage}{\bsf@pagebeforesub}}}%
    \addtocontents{nav}{\protect\headcommand{%
        \protect\beamer@documentpages{\bsf@pagebeforesub}}}%
```

The number of the last frame of the main part is stored in `\bsf@totalframenumber`. And `\beamer@writeslidentry` is replaced by `\beamer@writeslidentry@navbar`.

```
    \edef\bsf@totalframenumber{\the\c@framenumber}%
    \let\beamer@writeslidentry\beamer@writeslidentry@navbar
```

Before loading the `.sfr` file, it is closed and the flags for the navigation bar are initialized. The flag for `\ifsubframe` is set to true, because from now on there are only subframes.

```
    \immediate\closeout\bsf@sfrout
    \@bsf@firstparttrue
    \@bsf@firstsectiontrue
    \@bsf@firstsubsectiontrue
    \@bsf@subframetrue
    \input{\jobname.sfr}
```

After all subframes are typeset, the final `\bsf@...pages` entries are written to the `.nav` file.

```
    \@tempcnta\c@page\advance\@tempcnta -1\relax
    \if@filesw
      \immediate\write\@auxout{\string\@writefile{nav}%
        {\noexpand\headcommand{\noexpand\bsf@partpages{\bsf@partfirstsubframepage}%
          {\the\@tempcnta}{\bsf@partstartpage}{\bsf@pagebeforesub}}}}%
      \immediate\write\@auxout{\string\@writefile{nav}%
        {\noexpand\headcommand{\noexpand\bsf@sectionpages{\bsf@sectionfirstsubframepage}%
          {\the\@tempcnta}{\bsf@sectionstartpage}{\bsf@pagebeforesub}}}}%
      \immediate\write\@auxout{\string\@writefile{nav}%
        {\noexpand\headcommand{\noexpand\bsf@subsectionpages{\bsf@subsectionfirstsubframepage}%
          {\the\@tempcnta}{\bsf@subsectionstartpage}{\bsf@pagebeforesub}}}}%
      \immediate\write\@auxout{\string\@writefile{nav}%
        {\noexpand\headcommand{\noexpand\bsf@documentpages{\the\@tempcnta}}}}%
```

Finally, the command to disable the remaining `\beamer@...pages` entries is written to the `.nav` file.

```
      \immediate\write\@auxout{\string\@writefile{nav}%
        {\noexpand\headcommand{\noexpand\bsf@disablenaventries}}}%
    \fi
  \fi
}
```

8.5.8 At Begin and End of the Document

At the beginning of the document, first the `.sfp` file is loaded. The `.sfr` file is only opened in append mode.

```
\AtBeginDocument{%
  \InputIfFileExists{\jobname.sfp}{}{}%
  \if@bsf@append
    \immediate\openout\bsf@sfrout\jobname.sfr\relax
```

To reenable the `\beamer@...pages` entries after the first page, the command `\bsf@enablenaventries` is written as the first entry to the `.nav` file.

```
    \if@filesw
      \immediate\write\@auxout{\string\@writefile{nav}%
        {\noexpand\headcommand{\noexpand\bsf@enablenaventries}}}%
    \fi
  \fi
}
```

At the end of the document in append mode first the `.sfp` file is checked. After that `\bsf@totalframenumber` is initialized if necessary, i.e. if `\appendsubframes` was omitted in the source.

```
\AtEndDocument{%
  \if@bsf@append
    \bsf@checksfp
    \@ifundefined{bsf@totalframenumber}{%
      \edef\bsf@totalframenumber{\the\c@framenumber}}{}%
    \if@filesw
```

The definition for `\inserttotalframenumber` is written to the `.nav` file a second time, thus overwriting the first definition written by Beamer. Here it contains the number of frames of the main part.

```
      \immediate\write\@auxout{\string\@writefile{nav}%
        {\noexpand\headcommand{%
          \noexpand\def\noexpand\inserttotalframenumber{\bsf@totalframenumber}}}}
```

A stream for the for the `.sfp` file is defined and the file is opened, so LaTeX can write it, when loading the `.aux` file at the end of the document.

```
      \newwrite\tf@sfp
      \immediate\openout\tf@sfp\jobname.sfp\relax
    \fi
  \fi
```

Regardless of the mode, the definition of `\inserttotalframenumberwithsub` is written to the `.nav` file, so it can be used in themes in both modes.

```
\if@filesw
  \immediate\write\@auxout{\string\@writefile{nav}%
    {\noexpand\headcommand{%
      \noexpand\def\noexpand\inserttotalframenumberwithsub{\the\c@framenumber}}}}
\fi
}
```

8.5.9 Contitional Execution

`\ifappend` The macro `\ifappend` simply expands to its first argument in append mode and to its second in embed mode.

```
\newcommand{\ifappend}[2]{\if@bsf@append #1\else #2\fi}
```

`\ifsubframe` The macro `\ifsubframes` simply expands to its first argument in subframes and to its second in normal frames.

```
\newcommand{\ifsubframe}[2]{\if@bsf@subframe #1\else #2\fi}
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

`\bsfrestorepart` 137, 152
`\bsfrestoresection` 142, 177
`\bsfrestoresubsection` 147, 213
`\bsfsubframepages` 67, 73

D

`\dohead` 91

E

`embed` (option) *3*, 29
environments:
  `lastframe` *5*, 334
  `subframe` *4*, 277

H

`\hyperlinkframestartnext` 105
`\hyperlinkslidenext` 102

I

`\if@bsf@append` 15, 135, 277, 338, 381, 390, 408
`\if@bsf@firstline` 21, 293
`\if@bsf@firstpart` 22, 153
`\if@bsf@firstsection` 23, 178
`\if@bsf@firstsubsection` 24, 214
`\if@bsf@miniframes` . . 16, 280, 289, 301
`\if@bsf@nosfnum` 20, 92
`\if@bsf@nosubsection` 25, 190
`\if@bsf@prevnosubsection` 26, 224
`\if@bsf@subframe` 18, 409
`\ifappend` *4*, 408
`\ifsubframe` *6*, 409
`\inserttotalframenumber` . *5*, 276, 397
`\inserttotalframenumberwithsub` *5*, 276, 405

L

`lastframe` (environment) *5*, 334

N

`nominiframes` (option) *4*, 31

O

options:
  `append` *3*, 30
  `embed` *3*, 29
  `nominiframes` *4*, 31

P

`\PackageWarningNoLine` 93
`\part` 132, 136
`\part@orig` 132, 139
`\partentry` 86

S

`\section` 133, 141
`\section@orig` 133, 144
`\sectionentry` 87
`subframe` (environment) *4*, 277
`\subsection` 134, 146
`\subsection@orig` 134, 149
`\subslideentry` *6*, 54, 72, 89

T

`\tf@sfp` 398, 399